VITA

Q1

To Handel exception in C# you must use

A. Try catch block ===answer
B. Only try
C. Try – finally
D. None

Q2 All Exceptions derived from

A. Exception class ===answer
B. Application exception
C. System Exception

Q3.

A .An anonymous method cannot access ref or out parameters of the defining method.

B An anonymous method cannot have a local variable with the same name as a local

A. Only B is true
B. only A is true
C. none
D. both statements are true ===answer
E.

Q4

```
delegate void CountIt(int end);

class AnonMethDemo3 {
        static void Main() {
                int result;

CountIt count = delegate (int end) {
                int sum = 0;
                        for(int i=0; i <= end; i++) {
                        Console.WriteLine(i);
                        sum += i;
                        }
                return sum; // return a value from an anonymous method
        };
result = count(3);
Console.WriteLine("Summation of 3 is " + result);
}
}
```

A. 6===answer
B. 0
C. Error
D. None

Q5

```
delegate int CountIt(int end);

class AnonMethDemo3 {
        static void Main() {
                int result;

CountIt count = delegate (int end) {
                int sum = 0;
                        for(int i=0; i <= end; i++) {
                        Console.WriteLine(i);
                        sum += i;
                        }
                return sum; // return a value from an anonymous method
        };
result = count(3);
Console.WriteLine( result);
}
}
```

A.6 ==answer

B.none

c. 1,2,3,

Q6.

```
 delegate int addition(int x, int y);

  class myclass

  {

    public int add(int p, int q)

    {

      return p + q;

    }
```

```
        public int mul(int p, int q)
        {
            return p * q;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            myclass m = new myclass();
           addition a =delegate(int p,int q){int r; r=p+q; return r;};
            a += delegate(int p, int q) { int r; r = p * q; return r; };
                Console.WriteLine(a.GetInvocationList().Length);
               int invo = a(3, 5);
               Console.WriteLine(invo);
                Console.ReadLine();
        }
    }
}
```

A.15 ---answer

b.15,8

c.Error

d. none

Q7

```
delegate int Incr(int v);
```

VITA

```
class SimpleLambdaDemo {


  static void Main() {


   Incr incr = count => count + 2;

   int x = -5;

    while(x <= 0) {

      Console.Write(x + " ");

      x = incr(x); // increase x by 2

    }

}

}
```

A. -5,-3,-1 ===answer
B. None
C. 5,3,1,
D. Error

Q8 deligate for this lambda expression

```
n => n % 2 == 0
```

A. delegate true deli();
B. deligate bool deli(); ==answer
C. deligate int deli();
D. none

Q9

```
Using system;

Class myclass

{ public static void Main()

{

    IsEven isEven = n => n % 2 == 0;
```

VITA

```
    // Now, use the isEven lambda expression
    Console.WriteLine("Use isEven lambda expression: ");
    for(int i=1; i <= 3; i++)
      if(isEven(i))
Console.WriteLine(i + " is even."); } }
```

A.2---answer

B none

C 1,2,3

d.Error

Q 10

Data written before => is known as

A. input parameter ==answer

b. output parameter

c. represent return value

d. None